MATH465 Environmental Epidemiology

Lab Session 1

January 29,2007

Objectives of the Day 1

- Be aware of various R libraries which can be used for analysing spatial data
- Be familiar with plotting functions/options in `R`
- Be familiar with basic plotting functions in `splancs`

In the first three labs, we will use `R` software with two add-on packages: *splancs, spatialkernel*. *splancs* is one of the R libraries specifically designed for analysing spatial point pattern data. For other `R` libraries that handle various types of spatial data, they are grouped as one of the *CRAN Task Views* at
http://cran.r-project.org/src/contrib/Views/Spatial.html

Getting Started

Log into `cent1`, then `penguin`. It is useful to create the following two directories under your home directory,

```
% mkdir Rlibs
% mkdir m465
```

To start using `R`

```
% cd m465
% R &
```

Alternatively, open an `emacs` window. In `emacs`, press `Esc+X` simultaneously. On the bottom of the screen you should get `M-x`. Type in `R`, then press `enter`. You should have initiated R and got a '>' prompt.

To Quit `R`, type 'q()'. Then `R` will ask you

```
> q()
Save workspace image?  [y/n/c]:
```

By typing 'y', the objects you have created in this session will be stored in the file '.RData' under 'm465' directory. So if you log into `R` in the future, those objects you have created today will be restored automatically.

`splancs` should have been installed in the machine. To access `splancs` in a `R` session, we use one of the following two commands,

```
> library(splancs)
```
or
```
> require(splancs)
```

In general, if you want to install/update a library from internet to your own machine, type

```
> install.packages('splancs',dep=T,lib="~/Rlibs/")
```

(make sure your computer is connected to the internet!). For some packages to be installed/run properly, they depend on other packages to be installed first. Set "`dep=T`" and the system will make sure those packages are installed prior to the current required one.

If the above doesn't work for some reason, you can always download/install it manually from the internet. You can also find the user manual there. Go to
`http://www.r-project.org/`
Then select `Download CRAN` from the left hand side of the page.
Select a Mirror.
Then click `Contributed extension packages`.
You will find `splancs` there.

After successfully loading a package into `R`, if we are interested in what functions are included in such a package, use

```
> ls(pos=2)
```

To find out more about a function, try

`> help(install.packages)` or `> ?install.packages`

1 Plotting

Visually examining the spatial data through plotting can provide us the first impression of a data set.

While doing so, it is very important to preserve the correct scales of the data, i.e. to set "`asp=1`" allows one unit in x-axis/y-axis have the same length.

```
> x <- runif(200,0,1)
> y <- runif(200,0,2)
> plot(x,y)
> plot(x,y,asp=1)
```

To produce/save a graph for future usage (for example, to incorporate it into a Latex report),

```
> postscript(file='myfirstgraph.eps',print.it=F,onefile=T)
> plot(x,y,asp=1)
> dev.off()
```

Although it looks as if nothing has happened in the current `R` session, a file `myfirstgraph.eps` should be created under your m465 directory. This file can be opened via `gostview`.

Above plotting use only `R`'s standard commands.

Alternatively, we can use some functions in `splancs` to do the same things,

```
> poly1<-cbind(c(0,1,1,0),c(0,0,2,2))
> pts1<-as.points(x,y) #use the points we just created
> pts2<-csr(poly1,200) #use function to generate a complete spatial randomness data
> par(mfrow=c(1,2))
> polymap(poly1,lty=1,lwd=2)
> pointmap(pts1,pch=19,add=T,cex=0.2)

> polymap(poly1,lty=1,lwd=1)
> pointmap(pts2,pch=2,add=T,cex=1,col='red')
```

The reason we don't need to set `asp=1` in `polymap()` is because it has been set as a default in such a function.

One issue is how to display the plots nicely (for visual effect). So be adventurous to change the options specified here.

The boundary will affect how we visualize data.

```
> poly2<-cbind(c(0,6,6,0),c(0,0,6,6))
> polymap(poly2,lty=1,lwd=2)
> polymap(poly1,lty=3,lwd=2,add=T)
> pointmap(pts1,pch=19,add=T,cex=0.2)
```

If we mistakenly treat our study region as the solid big box (`poly2`), we will report a cluster on the bottom left corner. However, if we correctly define the study region as the dashed small box, we know that these are just some spatially random points! This is a rather artificial example. But we try to make a point that sometimes in a small region a point pattern may look one way. When the scale changes, the same data may look another way. So always be cautious in what you are doing!

Sometimes we need to define our own boundary. Type `getpoly()`, then use the left mouse button to click on the graphics window where you want the vertices of the polygon to be. When you finish, don't try to click on the first vertex again to join the box. Click on the middle button of the mouse and let the program do it for you.

```
> mypoly <- getpoly()
Enter points with button 1
Finish with button 2
Don't try to join the polygon up - it is done for you.
```

Using `pip()` function we can extract the points which lie within the defined region.

```
> mypts<-pip(pts1,mypoly)
> polymap(poly1,lty=1,lwd=1)
> polymap(mypoly,lwd=2,add=T)
> points(mypts,pch=19)
```

Note that so far in this lab we only plot the locations of the data, and assume these locations are of scientific interest. As we have learnt from the class, this is only one type of spatial statistics, namely point processes. In another kind of study, we may be interested in some attributes attached

to a location. In that case we may be interested in displaying not just the data locations, but also the measures attached to those locations. However, `splancs` is not designed to handle such data and we will leave this to another course (MATH460: Geostatistics).

Task 1 Reproduce the case and control plots from Childhood leukaemia data in Humberside. Three data sets: (1) leukaemia.cases.d (2) leukaemia.controls.d (3) leukaemia.poly.d can be download from http://www.maths.lancs.ac.uk/∼diggle/pointpatterns/Datasets/
To read data into `R`, use `read.table()`.

Note that you should multiply all cases/control/poly coordinates by 10000 (because they have been rescaled from the original scale).

For the UK map, follow the instruction on
http://www.maths.lancs.ac.uk/∼rowlings/Geog/britIsles/

To add the UK map to a plot, use the command

```
> britIsles(add=T)
```

Task 2 Reproduce the lung and larynx cancers' plots. Go to the same site above and download the following files: "incinerator.larynx.d","incinerator.lung.d","incinerator.location.d".

Task 3 Check
http://cran.r-project.org/src/contrib/Views/Spatial.html to see what other `R` resources are on offer for spatial data analysis, if you have not yet done so.

MATH465 Environmental Epidemiology

Lab Session 2

January 30,2007

Objectives of the Day 2

- Be able to simulate various point patterns.
- Understand/use K function to conduct spatial clustering test with aid from `splancs`
- Use K function in `splancs` to compare two groups, to decide whether or not one group is more clustered than the other.

2 Simulation point patterns

Generate Homogeneous Poisson Processes (HPP) with $\lambda(x) = 400$ per unit square

Call `splancs`. To define a polygon to simulate data, use the commands we have learnt from Lab 1. Don't define `mypoly` to be too small.

```
> poly1<-cbind(c(0,1,1,0),c(0,0,2,2))
> mypoly <- getpoly()
```

Alternatively, if you just want a unit square polygon,

```
> mypoly<-cbind(c(0,1,1,0),c(0,0,1,1))
```

```
> lambda<-400  #intensity
> sizeofmypoly<-areapl(mypoly)
> npts1<-rpois(1,lambda*sizeofmypoly) # no of points in the area~Poisson
> hpp.pts1<-csr(mypoly,npts1) # where are they are uniformly distributed

> plot(mypoly,type='n',asp=1)
> polymap(mypoly,add=T)
> points(hpp.pts1)
```

Alternatively, we can generate data on a bigger area, then throw away the points outside the polygon. The reason we may want to do so is because if the study region happens to be an awkward shape, it is easier to generate data on a bigger and well defined region first, then keep the data in the study region.

```
> sizeofpoly1<-areapl(poly1)
> npts2<-rpois(1,lambda*sizeofpoly1) # no of points in the area~Poisson
> hpp.pts<-csr(poly1,npts2)
> hpp.pts2<-pip(hpp.pts,mypoly)

> plot(poly1,type='n',asp=1)
> polymap(poly1,add=T)
> polymap(mypoly,add=T)
> points(hpp.pts2,col='red')
> points(hpp.pts,pch=19,cex=0.2)
```

This graph also explains why the *edge effect* is important when we handle the spatial data. For example, when we use our simulated data (`hpp.pt2`) to estimate K function, we need to empirically count the pairs for which the inter-point distance is less than a certain distance. If we neglect the fact that there are points outside `mypoly` (do not take edge correction into account), we will underestimate those near the edge.

Generate Inhomogeneous Poisson Processes (IPP) with $\lambda(x) = 10 + 100 * (xaxis) + 200 * (yaxis)$.

To see how $\lambda(x)$ looks like, we plot it on a fine grid.

```
> int<-function(x,y){10+100*(x)+200*(y)}

> x<-seq(0,1,l=51)
> y<-seq(0,2,l=101)
> z<- matrix(nrow=51,ncol=101)
> for (i in 1:51){
> z[i,]<-int(x[i],y)}

> par(mfrow=c(1,3))
> persp(x,y,z)
> image(x,y,z,asp=1)
> contour(x,y,z,asp=1)
```

Again, there are various ways of doing such a simulation. Here, we generate data with the right intensity for a larger area first. Then we keep only the data in `mypoly`.

```
> npts<-rpois(1,520)  #520 is from intergrate the intensity over poly1 by hand
> ipp.pts1<-matrix(0,nrow=npts,ncol=2)
> maxz<-int(1,2) #the maximum intensity in poly1
> for (i in 1:npts){
>  keep<-1
>  accp<-0
>  while(keep>accp){#we accept new point with probability proportion to intensity
```

```
>    simx<-runif(1,0,1)
>    simy<-runif(1,0,2)
>    keep<-runif(1,0,1)
>    accp<-int(simx,simy)/maxz
>  }
>  ipp.pts1[i,]<-c(simx,simy)
> }

> ipp.pts1<-as.points(ipp.pts1)
> my.ipp.pts1<-pip(ipp.pts1,mypoly) #only keep the point inside mypoly
> polymap(poly1)
> polymap(mypoly,add=T)
> points(my.ipp.pts1,col='red')
```

Another way of doing this is to simulate a homogeneous Poisson process with the maximum intensity (maxz=520 in this case) $\lambda(x)$ can get. Then for each simulated point (x', y'), keep it with probability `int(x',y')/maxz`.

```
> sizeofpoly1<-areapl(poly1)
> npts<-rpois(1,maxz*sizeofpoly1) # no of points in the area~Poisson
> ipp.pts2<-csr(poly1,npts)
> keep<-runif(npts,0,1)
> ind <-keep<(int(ipp.pts2[,1],ipp.pts2[,2])/maxz)
> ipp.pts2<- ipp.pts2[ind,]
> my.ipp.pts2<-pip(ipp.pts2,mypoly)
> polymap(poly1)
> polymap(mypoly,add=T)
> points(my.ipp.pts2,col='red')
```

You may like to plot the simulated data side by side for comparison. They should look similar.

```
> par(mfrow=c(1,2))
> polymap(poly1)
> points(ipp.pts1,pch=19,cex=0.2)
> polymap(poly1)
> points(ipp.pts2,pch=19,cex=0.2)
```

Task 1 Simulate Poisson cluster processes (PCP) on an unit square. Intensity of parents is 15 per unit square. No of children ~Poisson(30), the locations of the offspring follow bivariate normal distributions with marginal standard deviation of 0.04.

`R` code can be found on the lab solution. Give it a go before checking the answer.

From this simulation we have seen some clear patterns of clusters. We have also seen that both inhomogeneous Poisson processes or Poisson cluster process are capable to produce aggregated data. Different models may generate similar point patterns. In a real life problem if we only have one data set, it is impossible to distinguish which model the data come from.

Quite often, to study such clusters (e.g. what caused the data cluster together?) is an interesting scientific question of its own right.

3 Testing for spatial clustering

We have seen the data and created various plots. One natural question to ask next is: are they completely spatially random? We can use the K function for such a test. We will compare $\hat{K}(s)$ calculated from data to $\pi * s^2$, the theoretical value under HPP assumption.

The particular function we are using is `khat()`. We will use the simulated data (from HPP) we have just generated for a test run.

```
> s<-seq(0,0.15,by=0.01)
> kcsr1<-khat(hpp.pts1,mypoly,s)
> plot(s,pi*s^2,type='n',ylab='k(s)',lty=1)
> lines(s,pi*s^2)
> lines(s,kcsr1,col='red')

> kcsr2<-khat(hpp.pts2,mypoly,s)
> plot(s,pi*s^2,type='n',ylab='k(s)',lty=1)
> lines(s,pi*s^2)
> lines(s,kcsr2,col='red')
```

Two lines are almost overlapping! This is what we should expect since we simulate our data under HPP/CSR-the benchmark model we are comparing to. You may like to consider a different upper range for s (I decided mine based on the shorter side of `mypoly`)

Now we do the same thing on the simulated data generated from IPP,

```
> par(mfrow=c(1,2))
> kipp1<-khat(my.ipp.pts1,mypoly,s)
> plot(s,pi*s^2,type='n',ylab='k(s)',lty=1)
> lines(s,pi*s^2)
> lines(s,kipp1,col='red')

> kipp2<-khat(my.ipp.pts2,mypoly,s)
> plot(s,pi*s^2,type='n',ylab='k(s)',lty=1)
> lines(s,pi*s^2)
> lines(s,kipp2,col='red')
```

It is difficult to tell the difference between two lines by your raw eyes. So we firstly subtract all $\hat{K}(s)$ by πs^2; secondly, we do Monte Carlo tests of CSR and add 95% tolerance limits as follows,

```
> dat<-csr(mypoly,nrow(my.ipp.pts1))  #generate 200 datasets under H0 of CSR
```

```
> mat<-khat(dat,mypoly,s)
> for(i in 2:200){
> dat<-csr(mypoly,nrow(my.ipp.pts1))
> mat<-rbind(mat,khat(dat,mypoly,s))
> cat(i, ' ')
> }

> tollimit<-apply(mat,2,quantile,prob=c(0.025,0.975))

> plot(s,kipp1-pi*s^2,type='n',ylab='k(s)',lty=1,ylim=c(-0.002,0.007))
> lines(s,kipp1-pi*s^2,type='l',ylab='k(s)-pi*s^2')
> abline(h=0)
> lines(s,tollimit[1,]-pi*s^2,lty=2)
> lines(s,tollimit[2,]-pi*s^2,lty=2)

> plot(s,kipp2-pi*s^2,type='n',ylab='k(s)',lty=1,ylim=c(-0.002,0.007))
> lines(s,kipp2-pi*s^2,type='l',ylab='k(s)-pi*s^2')
> abline(h=0)
> lines(s,tollimit[1,]-pi*s^2,lty=2)
> lines(s,tollimit[2,]-pi*s^2,lty=2)
```

What can you conclude from this graph? To satisfy your own curiosity, you may like to repeat the same Monte Carlo test on the other simulated data (both HPP and PCP) we have just generated. We would expect Khat from HPP will fall in the envelope at most of the distances.

By using simulated data in the above exercise, we learn what the K function should look like. The empirical K will fall above the upper limit if the point pattern is a cluster one, within the envelope if PP is at random, below the envelope if PP is inhibited.

Note that while doing the above exercise, we should consider the distance/range of s. When s is large, $K(s)$ always involves large variability (can tell from the horn shape of the tolerance envelope).

When we compare $\hat{K}(s_0)$ at distance s_0 to the Monte Carlo envelope, we are doing a testing. So when we try to look/explain the whole trend, we are doing more than one test (multiple testings) in our mind! But don't worry too much. This is just an exploratory data analysis and is to provide us some insights about the data.

4 Case Control Study: Is one group more clustered than the other?

In the previous section we have seen how one data set can be compared against the benchmark model- CSR. If we have two data sets, we can compare them to each other and decide if they are quantitatively different- Is one group more clustered than the other? What the spatial patterns look like? Do patterns from different groups vary spatially? We will introduce a method to test the first question and leave the other questions to the next Lab.

Case control study is a type of epidemiological study (others include cohort study, experimental study such as clinical trials). An important issue is how to select a control group carefully in order to answer the scientific question we have in mind.

The main purpose of this lab is to demonstrate the statistical methods which can be used to answer

some questions that may raise from such a study. For this demonstration purpose, we will use the simulated data we have just generated as either CS or CN.

In the first example, we simulate HPP data twice and assign them to be the case/control.

The statistics we are using is the difference between two K functions $D(s) = K_{cs}(s) - K_{cn}(s)$. Under the null hypothesis of no spatial clustering, they are comming from the same population. So we can pull all the data together, randomly relabel them as cases or controls, then recalculate K (s)/ D(s). If we do this plenty of times, we can find the empirical upper and lower limits. These are the Monte Carlo tests (which are simulation-based) that we are going to do first. The function we are using is `Kenv.label()`.

```
> case<-hpp.pts1
> control<-hpp.pts2
> dhat<-khat(case,mypoly,s)-khat(control,mypoly,s) #Calculate D(s)
> plot(s,dhat,type='l',ylab='D(s)')
> abline(h=0)

#Random labelling
> dlim<-Kenv.label(case,control,mypoly,200,s)
> dhatlower<-dlim$lower
> dhatupper<-dlim$upper

> plot(s,dhat,type='l',ylab='D(s)',ylim=c(-0.01,0.01))
> abline(h=0)
> lines(s,dhatlower,lty=2)
> lines(s,dhatupper,lty=2)
```

An alternative way of adding the envelope is by calculating the standard error for $D(s)$. Then we can plot 2 times the standard error, i.e. $\approx 95\%$ confidence limits. In `splancs` it can be done using the `secal()` function.

```
> dhatse<-secal(case,control,mypoly,s)
> lines(s,-2*dhatse,lty=3)
> lines(s,2*dhatse,lty=3)
```

Since we know both our artificial CS/CN are from the same HPP model, we know neither group is more crowded than the other. Hence, most of $\hat{D}(s)$ should lie comfortably inside the envelope.

Task 2 Repeat the above exercise but change the case to IPP or PCP. What do you conclude from the graphs?

```
> case<-my.ipp.pts2
> case<-my.pcp.pts2
```

To "summarize" $D(s)$, Diggle and Chetwynd (1991) proposed

$$D = \int_0^{s_0} \hat{D}(s)/\sqrt{v(s)}ds$$

(as seen in the class). We are going to see how this test statistics can be useful. Note that we use numerical integration here.

```
> case<-hpp.pts1
> control <-hpp.pts2
> all <- rbind(case, control)

> nocase <- nrow(case) # number of cases
> nototal <- nrow(all) # total

#For observed data
> sterror <- secal(case,control,mypoly,s)
> dhat <- khat(case,mypoly,s) - khat(control,mypoly,s)
> dobserved <- sum(dhat[-1]/sterror[-1])

#random relabel case/control 199 times for MC test
> d <- NULL

> for(i in 1:100){
>  if (i%%10==0)
>    {cat(i, '\ ')}

> indcas <- sample(c(1:nototal),size=nocase)
> case <- all[indcas,]
> control <- all[-indcas,]
> sterror <- secal(case,control,mypoly,s)
> dhat <- khat(case,mypoly,s) - khat(control,mypoly,s)
> d <- c(d,sum(dhat[-1]/sterror[-1]))
>}
```

To see how D looks like,

```
> hist(d,probab=T)
> abline(v=dobserved,col=2)
```

There is a chance your `dobserved` falls outside the current plotting range. If so, set `xlim` in `hist()` and re-draw the plot.

P-value is calculated by ranking the observed D statistics among other D statistics under random permutation of H_0.

```
> pvalue <- length(d[d>dobserved])/length(d)
```

Task 3 Repeat the above test on cases from PCP and controls from HPP.

We should expect a very tiny P-value here since the cases are much more clustered than the controls.

Task 4 Use childhood leukaemia data to conduct a Monte Carlo test on D statistics. Report your p-value. Compare your conclusion to Peter's (on Note p27).

MATH465 Environmental Epidemiology

Lab Session 3

Feburary 1,2007

Objectives of the Day 3

- Use Kernel smoothing to produce smoothed intensity map
- Learn about the `spatialkernel` package

5 Kernel estimation

Visual inspection of the points can be difficult and misleading. One tool that helps is to produce a smooth (intensity) map via kernel smoothing. It is a non-parametric method done by putting a small "weight"(kernel) on each data point, then summing up these weights over the whole area. Two factors: bandwidth size and kernel function (gaussian, quadratic,quartic, epanechnikov, etc.) need to be considered by a user. Theoretical work shows that carefully tunning the bandwidth is more important than choosing a kernel.

We use `kernel2d()` to generate the map. There are some automatic methods of choosing bandwidth. (`mse2d()`) is designed to pick h by minimising the integrated mean square error (integrated over the whole space).

We will try this on the IPP data. Use the code from yesterday to generate an IPP data set with $\lambda(x) = 10 + 100 * (xaxis) + 200 * (yaxis)$. Set `mypoly` to be unit square.

```
> case <- my.ipp.pts2
> temp<-mse2d(case,mypoly,nsmse=50,range=0.25)
> plot(temp$h,temp$mse,type='l')
> besth<-temp$h[temp$mse==min(temp$mse)]
```

Take a look at the bandwidth computer selected for us (mine was 0.195 but yours may be different). Then use such bandwidth to estimate the intensity. Note that only the quartic kernel is implemented in `splancs`.

```
> ipp.int<-kernel2d(case,mypoly,besth,50,50)
```

Now we will plot $\hat{\lambda}(s)$ and the true $\lambda(s)$ (since we are using the simulated data, we know the true intensity!) side by side. They are qualitatively similar with the increasing intensity (50-300) from left-bottom corner to right-top corner.

```
> par(mfrow=c(1,2))
```

```
> plot(mypoly,xlim=c(0,1),ylim=c(0,2),type='n',asp=1)
> polymap(mypoly,axes=F,add=T)
> image(ipp.int,add=T)
> contour(ipp.int,add=T)
> points(case,pch=19,cex=0.3)
> title(paste('IPP:Kernel estimate use Quartic kernel, h=',besth))

> x<-seq(0,1,l=51)
> y<-seq(0,1,l=101)
> z<- matrix(nrow=51,ncol=101)
> for (i in 1:51){
> z[i,]<-int(x[i],y)}
> image(x,y,z,asp=1)
> contour(x,y,z,asp=1,add=T)
> title(paste('Case:true intensity',besth))
```

To see the effect of changing h on the appearance of $\hat{\lambda}(s)$, pick two extra h arbitrarily- one larger and one smaller than the current `besth`.

```
> ipp.int.1<-kernel2d(case,mypoly,besth,50,50)
> ipp.int.2<-kernel2d(case,mypoly,0.4,50,50)
> ipp.int.3<-kernel2d(case,mypoly,0.1,50,50)
```

```
> par(mfrow=c(1,3))
> plot(mypoly,type='n',asp=1)
> polymap(mypoly,axes=F,add=T)
> image(ipp.int.1,add=T)
> contour(ipp.int.1,add=T)
> points(case,pch=19,cex=0.3)
> title(paste('IPP:Kernel estimate use Quartic kernel, h=',besth))

> plot(mypoly,type='n',asp=1)
> polymap(mypoly,axes=F,add=T)
> image(ipp.int.2,add=T)
> contour(ipp.int.2,add=T)
> points(case,pch=19,cex=0.3)
> title('IPP:Kernel estimate use Quartic kernel, h=big')

> plot(mypoly,type='n',asp=1)
> polymap(mypoly,axes=F,add=T)
> image(ipp.int.3,add=T)
> contour(ipp.int.3,add=T)
> points(case,pch=19,cex=0.3)
> title('IPP:Kernel estimate use Quartic kernel, h=small')
```

As we can see, if h is big, the estimate is smooth; when h is small, the whole surface is rough.

Next, try the same procedure on the simulated data from PCP/HPP.

```
case <-hpp.pts2
case <- my.pcp.pts2
```

What did you notice? Did you notice the L-shape of MSE when the process is homogeneous? This is because the optimal (by the current selection criteria) $h \to \infty$ for HPP.

On the other hand (not shown here) we may have a point pattern with lots of coincident points at the same location (for example, a matched CS-CN study with siblings being selected as control, so their geo-labels are from the same household). Under this scenario, optial $h \to 0$.

You may like to create some artificial point patterns with plenty of coincident points. Then repeat the above procedure to verify the previous statement.

The above example is aiming to illustrate that the automatic selector can be handy, but it is not always the best choice. Since a map always has its scientific meaning (so does h), most of the time the scientist must have some "belief" how smooth an intensity map may/should be. We should use the results from a selector as a guide but always use our knowledge/mind to produce the final map!

6 Case Control study: Spatial variation in risk

We have learnt how to use kernel smoothing to estimate the intensity. To explain the spatial variation in risk, as seen in the notes, there are three approaches: density ratio, binary regression, and generalized additive model.

In this section we will introduce how binary regression can be done via another `R` library, namely `spatialkernel`. This library has some improved functions over `splancs`. In fact, it can handle not only binary kernel regression but *multivariate* spatial point patterns.

Again, the user manual can be found on the R-project web site. We encourage you to take a look of the summary provided by the author,
http://www.maths.lancs.ac.uk/∼zhengp1/spatialkernel/

```
> require(spatialkernel)
```

Here, we assign cases from the PCP model and the control group to be from the HPP model. Again, you can use yesterday's code to produce such data.

```
> case <- my.pcp.pts2
> control<-hpp.pts2

> plot(mypoly,type='n',asp=1)
> polygon(mypoly,add=T)
> points(control,col='red')
> points(case,col='blue',pch='x')
```

We pull all the data together and create a mark/label vector (`mks`). `cvloglk()` conducts the likelihood cross-validation as an automatic bandwidth selector. Note that, since the objective function is a log-likelihood function, we want to maximize it while looking for h (compare to the `mse2d()` which is an error function that we want to minimize).

```
> h1 <- seq(0.01, 0.25, by=0.01)
> all <- rbind(case, control)
> mks<-c(rep("case",length=nrow(case)),rep("control",length=nrow(control)))

> allcv <- cvloglk(all,mks,h=h1)$cv
> plot(h1, allcv, type="l")
> hmax <- h1[allcv==max(allcv)]
```

Here, a common h is selected for both CS/CN groups.

The following step may take a while. Try to read the help manual to see what the `spseg()` function does.

```
> sp <- spseg(all, mks, hmax, opt=3, ntest=100, poly=mypoly)

> brksuse <- pretty(range(sp$p, na.rm=TRUE), n=12)
> par(mfrow=c(1,2))
> plotphat(sp)
> plotphat(sp,cex=2,pch='.',breaks=brksuse,col=risk.colors(length(brksuse)-1))
> metre(0, 1.1,1, 1.3, lab=brksuse, col=risk.colors(length(brksuse)-1), cex=1)
> plotmc(sp,quan=c(0.025,0.975))
```

The first two plots show the type-specific probabilities. Since we only have case and control groups, these two plots complement each other (one shows $p(x)$, the other shows $1 - p(x)$).

Contour plots show the Monte Carlo test results. These results are quite obvious if you add the points onto the plots (by using `points()`).

The function `lambdahat()` in `spatialkernel` can estimate $\lambda(s)$.

```
> gridxy <- as.matrix(expand.grid(x=seq(0, 1, length=101), y=seq(0, 1, length=101)))
> lam <- matrix(NA,ncol=101,nrow=101)
> lam[1:nrow(gridxy)]
     <- lambdahat(case, hmax, gpts = gridxy[1:nrow(gridxy),], poly =mypoly)$lambda

> plot(0, 0, xlim=0:1, ylim=0:1, xlab="x", ylab="y", type="n",asp=1)

> brks <- pretty(range(lam),n=12)
> image(x=seq(0, 1, length=101), y=seq(0, 1, length=101),
         z=lam, add=TRUE, breaks=brks, col=risk.colors(length(brks)-1))
> polygon(mypoly)
> metre(0, 1.01,1, 1.1, lab=brks, col=risk.colors(length(brks)-1), cex=1)
```

Let's compare kernel estimates from both `splancs` and `spatialkernel` packages.

```
> int.splancs<-kernel2d(case,mypoly,hmax,50,50)

> par(mfrow=c(1,2))
> image(x=seq(0, 1, length=101), y=seq(0, 1, length=101),z=lam)
> title("From spatialkernel")
> image(int.splancs)
> title("From splancs")
```

They look similar, but not exactly the same. Why?

From here we have seen there are some functionalities implemented in more than one package. Another example is the implementation of the K function: `splancs`, `spatial`, `spatstat` (we do not introduce the latter two packages in this class but, again, they can be downloaded from the `R` web site). The usage of a function may be different from package to package, but the theoretical background behind a function will always the same. So as long as you read the manual/help page, to learn a new package won't be difficult!

7 Case Control Study:Generalized additive model

Generalized additive model is a semi-parametric model with the form

$$g(E(y_i)) = x_i\beta + s(z_i)$$

where $x_i\beta$ is a conventional parametric component of a linear predictor (in a CS/CN setting, this part can be the covariates we want to adjust for), and a non-parametric smoothing function $s(\ldots)$ of other covariates z_i. $g(\dot{)}$ is a link function (same as the ones used in traditional GLM). The penalized likelihood maximization is used for estimating the model parameters. This estimating method is done by adding an extra term into the likelihood function to penalize the "roughness" for each $s(\dot{)}$.

We will use `mgcv` package to do such analysis. (You may need to download this package first). For more theoretical background and details of how to use the `mgcv` library, refer to the book "Generalized Additive Models: AN introduction with R" , Chapman and Hall/CRC (2006) by Simon N Wood.

To explain how GAM model works, we use (partially) a pioneering epidemiology study by Dr. John Snow (1813-1858), who famously remove a pump handle in London to stop a cholera epidemic in 1854.

Download files "cholera.deaths", "cholera.pumps", and read "cholera.explain" from Peter's web site. (more interesting stories regarding John Snow can be found easily from the internet!!!)

Note that in the following example, we artificially create a homogeneous control group (cases are real). We also attach a fake age for each case/control.

```
> library(mgcv)
> deaths<-scan("deaths.txt")
> ndeaths<-length(deaths)/2
> deaths<-matrix(deaths,ndeaths,2,byrow=T)
> pumps<-scan("pumps.txt")
> npumps<-length(pumps)/2
> pumps<-matrix(pumps,npumps,2,byrow=T)
> poly<-cbind(c(8,18,18,8),c(6,6,17,17))
```

Task Write your codes to display deaths/pumps on a map!

Then we create a control group for such cases. We assume the population is homogeneously living in the defined area, and we take 2 controls per case.

```
> controls<-csr(poly,2*ndeaths)
> flags<-c(rep(1,ndeaths),rep(0,2*ndeaths))  #create a flag for cs/cn
```

```
> locations<-rbind(deaths,controls)
> deathsplus<-cbind(flags,rbind(deaths,controls), round(50+5*rnorm(3*ndeaths),1))#artificia
> names(deathsplus)<-c("flags","xcoord","ycoord","age")
> deathsplus<-as.data.frame(deathsplus)
```

We adjust for age in the GAM and let any residual effect be modelled by a smooth 2-D function. Make sure you know why a logit link is used. The plot shows `s(.)`. What do you notice? Check the significant test for each covariate added. Do you need to add those in the model?

```
> fit<-gam(flags~age+s(xcoord,ycoord),data=deathsplus,family=binomial(link=logit))
> summary(fit)
> plot(fit)
> points(pumps,pch='X',col='blue',cex=2)
```

We now add the distance from Broad Street pump as an explanatory variable.

```
> broadstreet<-matrix(pumps[7,],1,2)
> points(broadstreet,pch=19,col="blue")

> distance<-c(sqrt(outer(broadstreet[,1],deathsplus[,2],"-")^2+
> outer(broadstreet[,2],deathsplus[,3],"-")^2))

> deathsplus$distance<-distance
> fit2<-gam(flags~distance+s(xcoord,ycoord),data=deathsplus,family=binomial(link=logit))
> summary(fit2)
> plot(fit2)
> points(broadstreet,pch=19,col="blue")
```

Do you notice the contrasting apearance of fitted residual surfaces from two fits without and with distance from Broad Street pump?